I claim:

1. A method of optimizing the compiled code generated from high level computer programming languages which include loop constructs, the method comprising the steps:

(1) providing a loop code segment corresponding with a loop construct written in a high level programming language, in which the loop construct is executed a loop repetition number of times *n*;

(2) providing execution conditions required to cause execution of the loop construct the loop repetition number of times *n*;

(3) optimizing the loop code segment for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*;

(4) determining whether the consolidated code segment should be executed in preference to the corresponding code segments before said optimization; and

(5) if said determination is favourable, including the consolidated code segment in optimized code for a program written in the high level programming language.

2. A method of optimizing the compiled code generated from high level computer programming languages which include loop constructs, the method comprising the steps:

(1) providing a loop code segment corresponding with a loop construct written in a high level programming language, in which the loop construct is executed a loop repetition number of times *n*;

(2) providing a pre-loop code segment corresponding with programming instructions preceding the loop construct, and a post-loop code segment corresponding with instructions succeeding the loop construct;

(3) providing execution conditions required to cause execution of the loop construct the loop repetition number of times *n*;

(4) revising the pre-loop, loop and post-loop code segments to include the execution conditions; and

(5) optimizing the pre-loop, loop and post-loop code segments for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*;

(6) determining whether the consolidated code segment should be executed in preference to the corresponding code segments before said optimization; and

(7) if said determination is favourable, including the consolidated code segment in optimized code for a program written in the high level programming language.

3. The method as claimed in claim 1, wherein said determination involves a cost-benefit analysis to determine whether there the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

4. The method as claimed in claim 1, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

5. The method as claimed in claim 1, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

6. The method of claim 1, wherein said steps (1) to (5) are repeated a predetermined number of times *k*, for values of the loop repetition number *n* from 0 to *k*-1.

7. The method as claimed in claim 2, wherein said determination involves a cost-benefit analysis to determined whether there the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

8. The method as claimed in claim 2, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

9. The method as claimed in claim 2, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

10. The method of claim 2, wherein said steps (1) to (7) are repeated a predetermined number of times *k*, for values of the loop repetition number *n* from 0 to *k*-1.

11. A compiler for optimizing the compiled code generated from high level computer programming languages which include loop constructs, the compiler being embodied on a computer-readable medium, the compiler comprising:

(1) compiler code means for providing a loop code segment corresponding with a loop construct written in a high level programming language, in which the loop construct is executed a loop repetition number of times *n*;

(2) compiler code means for providing execution conditions required to cause execution of the loop construct the loop repetition number of times *n*;

(3) compiler code means for optimizing the loop code segment for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*;

(4) compiler code means for determining whether the consolidated code segment should be executed in preference to the corresponding code segments before said optimization; and

(5) compiler code means for including the consolidated code segment in optimized code for a program written in the high level programming language, if said determination is favourable.

12. A compiler for optimizing the compiled code generated from high level computer programming languages which include loop constructs, the compiler being embodied on a computer-readable medium, the compiler comprising:

(1) compiler code means for providing a loop code segment corresponding with a loop construct written in a high level programming language, in which the loop construct is executed a loop repetition number of times *n*;

(2) compiler code means for providing a pre-loop code segment corresponding with programming instructions preceding the loop construct, and a post-loop code segment corresponding with instructions succeeding the loop construct;

(3) compiler code means for providing execution conditions required to cause execution of the loop construct the loop repetition number of times *n*;

(4) compiler code means for revising the pre-loop, loop and post-loop code segments to include the execution conditions; and

(5) compiler code means for optimizing the pre-loop, loop and post-loop code segments for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*;

(6) compiler code means for determining whether the consolidated code segment should be executed in preference to the corresponding code segments before said optimization; and

(7) compiler code means for including the consolidated code segment in optimized code for a program written in the high level programming language, if said determination is favourable.

13. The compiler as claimed in claim 11, wherein said determination involves a cost-benefit analysis to determined whether there the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

14. The compiler as claimed in claim 11, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

15. The compiler as claimed in claim 11, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

16. The compiler of claim 11, wherein said steps (1) to (5) are repeated a predetermined number of times k, for values of the loop repetition number n from 0 to k-1.

17. The compiler as claimed in claim 12, wherein said determination involves a cost-benefit analysis to determined whether there the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

18. The compiler as claimed in claim 12, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

19. The compiler as claimed in claim 12, wherein said loop constructs includes any one

or more of the following loop constructs: for loops, while loops, repeat loops.

20. The compiler of claim 12, wherein said steps (1) to (7) are repeated a predetermined number of times k, for values of the loop repetition number n from 0 to k-1.